DOI:10.1145/3134599

Such inputs distort how machine-learning-based systems are able to function in the world as it is.

BY IAN GOODFELLOW, PATRICK MCDANIEL, AND NICOLAS PAPERNOT

Making Machine Learning Robust Against Adversarial Inputs

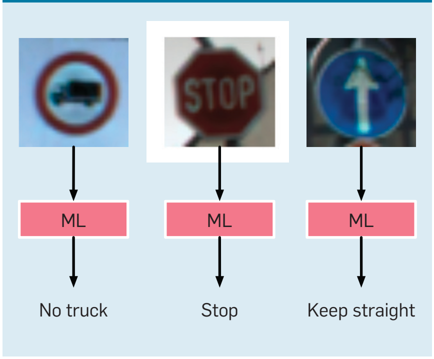
MACHINE LEARNING HAS advanced radically over the past 10 years, and machine learning algorithms now achieve human-level performance or better on a number of tasks, including face recognition,[31] optical character recognition,[8] object recognition,[29] and playing the game Go.[26] Yet machine learning algorithms that exceed human performance in naturally occurring scenarios are often seen as failing dramatically when an adversary is able to modify their input data even subtly. Machine learning is already used for many highly important applications and will be used in even more of even greater importance in the near future. Search algorithms, automated financial trading algorithms, data analytics, autonomous vehicles, and malware detection are all critically dependent on the underlying machine learning algorithms that interpret their respective domain inputs to provide intelligent outputs that facilitate the decision-making process of users or automated systems. As machine learning is used in more contexts where malicious adversaries have an incentive to interfere with the operation of a given machine learning system, it is increasingly important to provide protections, or "robustness guarantees," against adversarial manipulation.

The modern generation of machine learning services is a result of nearly 50 years of research and development in artificial intelligence—the study of computational algorithms and systems that reason about their environment to make predictions.[25] A subfield of artificial intelligence, most modern machine learning, as used in production, can essentially be understood as applied function approximation; when there is some mapping from an input x to an output y that is difficult for a programmer to describe through explicit code, a machine learning algorithm can learn an approximation of the mapping by analyzing a dataset containing several examples of inputs and their corresponding outputs. The learning proceeds by defining a "model," a parametric function describing the mapping from inputs to outputs. Google's image-classification system, Inception, has been trained with millions of labeled images.[28] It can classify images as cats, dogs, airplanes, boats, or more complex concepts on par or improving on human accuracy.

» key insights

- **Machine learning has traditionally been developed following the assumption that the environment is benign during both training and evaluation of the model; while useful for designing effective algorithms, this implicitly rules out the possibility that an adversary could alter the distribution at either training time or test time.**
- **In the context of adversarial inputs at test time, few strong countermeasures exist for the many attacks that have been demonstrated.**
- **To end the arms race between attackers and defenders, we suggest building more tools for verifying machine learning models; unlike current testing practices, this could help defenders eventually gain a fundamental advantage.**

Figure 1. Example machine learning task: traffic sign classification.[27]



Increases in the size of machine learning models and their accuracy is the result of recent advancements in machine learning algorithms,[17] particularly to advance deep learning.[7]

One focus of the machine learning research community has been on developing models that make accurate predictions, as progress was in part measured by results on benchmark datasets. In this context, accuracy denotes the fraction of test inputs that a model processes correctly—the proportion of images that an object-recognition algorithm recognizes as belonging to the correct class, and the proportion of executables that a malware detector correctly designates as benign or malicious. The estimate of a model's accuracy varies greatly with the choice of the dataset used to compute the estimate. The model's accuracy is generally evaluated on test inputs that were not used during the training process. The accuracy is usually higher if the test inputs resemble the training images more closely. For example, an object-recognition system trained on carefully curated photos may obtain high accuracy when tested on other carefully curated photos but low accuracy on photos captured more informally by mobile phone users.

Machine learning has traditionally been developed following the assumption that the environment is benign during both training and evaluation of the model. Specifically, the inputs x are usually assumed to all be drawn independently from the same probability distribution at both training and test time. This means that while test inputs x are new and previously unseen during the training process, they at least have the same statistical properties as the inputs used for training. Such assumptions have been useful for designing effective machine learning algorithms but implicitly rule out the possibility that an adversary could alter the distribution at either training time or test time. In this article, we focus on a scenario where an adversary chooses a distribution at test time that is designed to be exceptionally difficult for the model to process accurately. For example, an adversary might modify an image (slightly) to cause it to be recognized incorrectly or alter the code of an executable file to enable it to bypass a malware detector. Such inputs are called "adversarial examples"[30] because they are generated by an adversary.

The study of adversarial examples is in its infancy. Several algorithms have been developed for their generation and several countermeasures proposed. We follow with an overview of the foundations and advancements of this thriving research field and some predictions of where it might lead.

Figure 2. Example problem with two classes: the ideal decision boundary between the two models and the approximate boundary learned by the model.

Note that in higher dimensions, all examples are "close" to decision boundaries, as illustrated in this low-dimensional problem by the "pocket" of red class points included in the blue class.

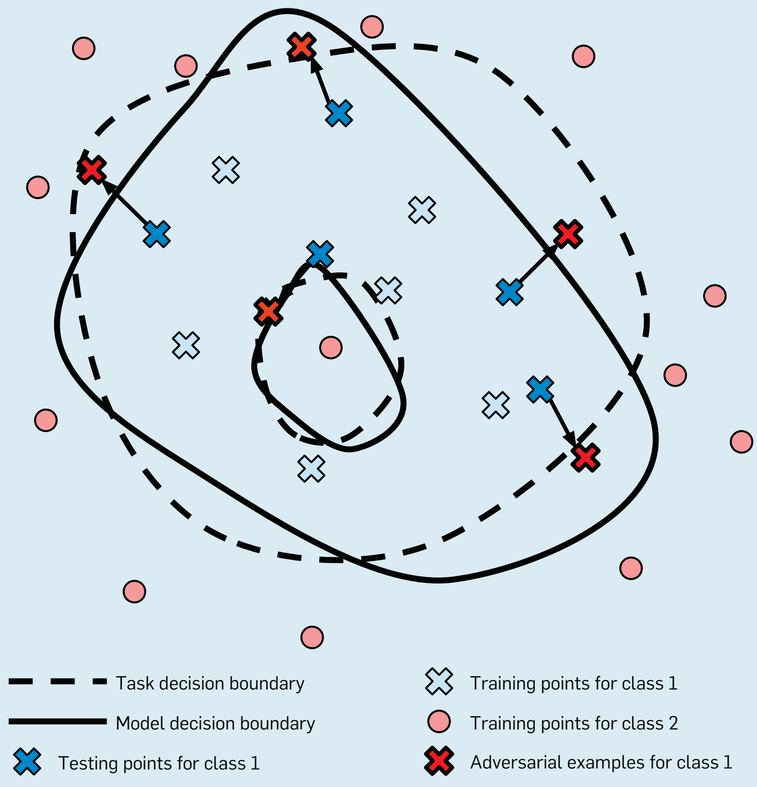
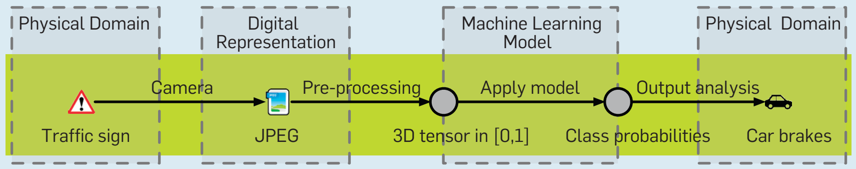

Task decision boundary
Model decision boundary
Testing points for class 1
Training points for class 1
Training points for class 2
Adversarial examples for class 1

Figure 3. Example machine learning pipeline in the context of autonomous driving.[23]



Machine Learning Systems in Adversarial Settings

To simplify our presentation in this article, we focus on machine learning algorithms that perform "classification," learning a mapping from an input x to a discrete variable y where y represents the identity of a class. As a unifying example, we discuss road-sign image recognition; the different values of y correspond to different types of road signs (such as stop signs, yield signs, and speed limit signs). Examples of input images and expected outputs are shown in Figure 1. Though we focus on image classification, the principles of adversarial machine learning apply to much more general artificial intelligence paradigms (such as reinforcement learning).[12]

Anatomy of a machine learning task. A machine learning algorithm is expected to produce a model capable of predicting the correct class of a given input. For instance, when presented with an image of a STOP sign, the model should output the class designating "STOP." The generic strategy adopted to produce such a model is twofold: a family of parameterized representations, the model's architecture, is selected, and the parameter values are fixed.

The architecture is typically chosen from among well-studied candidates (such as support vector machines and neural networks). The choice is made through either an exhaustive search or expert knowledge of the input domain. The chosen architecture's parameter values are fixed so as to minimize the model's prediction error over large collections of example pairs of inputs and expected outputs, the classes.

When this training phase is complete, the model can be used to predict the class of test inputs unseen during training. We can think of the classifier as defining a map of the input space, indicating the most likely class within each input region. The classifier will generally learn only an approximation of the true boundaries between regions for a variety of reasons (such as learning from few samples, using a limited parametric model family, and imperfect optimization of the model parameters), as shown schematically in Figure 2. The model error between the approximate and expected decision boundaries is exploited by adversaries, as explained in the following paragraphs.

The machine learning pipeline. Machine learning models are frequently deployed as part of a data pipeline; inputs to the model are derived from a set of preprocessing stages, and outputs of the model are used to determine the next states of the overall system.[23] For example, our running example of a traffic-sign classifier could be deployed in an autonomous vehicle, as illustrated in Figure 3. The model would be given as inputs images captured by a camera monitoring the side of the road and coupled with a detection mechanism for traffic signs. The class predicted by the machine learning model could then be used to decide what action should be taken by the vehicle (such as come to a stop if the traffic sign is classified as a "STOP" sign).

Attacking the system. As outlined earlier in this article, most machine learning models are designed, at least partially, based on the assumption that the data at test time is drawn from the same distribution as the training data, an assumption that is often violated. It is common for the accuracy of the model to degrade due to some relatively benign change in the distribution of test data. For example, if a different camera is used at test time from the one used to collect training images, the trained model might not work well on the test images. More important, this phenomenon can be exploited by adversaries capable of manipulating inputs before they are presented to the machine learning model. The adversary's motivation for "controlling" the model's behavior this way stems from the implications of machine learning predictions on consequent steps of the data pipeline.[23] In our running example of an autonomous vehicle, an adversary capable of crafting STOP signs classified as "yield" signs may cause the autonomous vehicle to disobey traffic laws and potentially cause an accident. Machine learning is also applied to other sensitive domains (such as financial fraud[3] and malware detection[1]) where adversarial incentives to have the model mis-predict are evident.

As is common in modeling the security of any domain, the domain of adversaries against machine learning systems can be structured around a taxonomy of capabilities and goals.[2,22,23] As reflected in Figure 4, the adversary's strength is characterized by its ability to access the model's architecture, parameter values, and training data. Indeed, an adversary with access to the model architecture and parameter values is capable of reproducing the targeted system on its own machine and thus operates in a "white-box" scenario. In addition to different degrees of knowledge about the model internals, adversaries may also be distinguished by their ability to submit inputs directly to the machine learning model or only indirectly through the data pipeline in Figure 3. For instance, Kurakin et al.[16] demonstrated that adversaries can manipulate a machine learning

Figure 4. A taxonomy of adversaries against machine learning models at test time.[22]

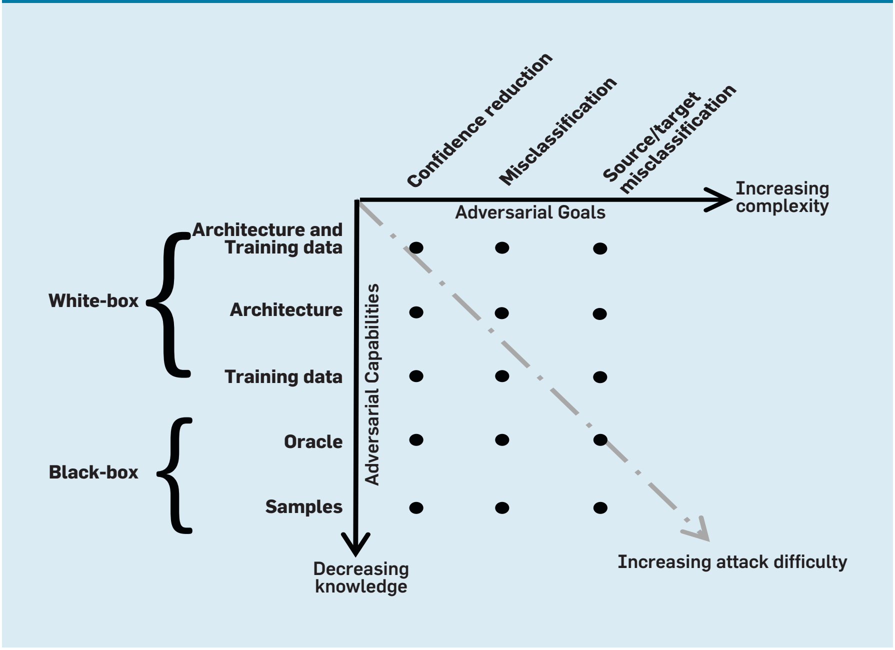


Figure 5. Original image (left) and adversarial image (right).

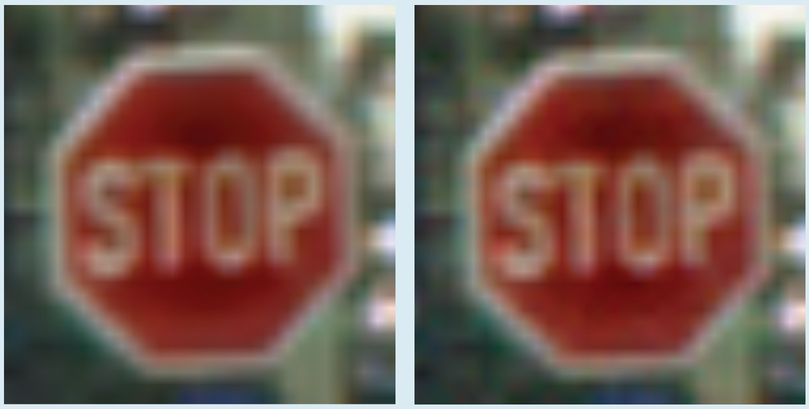


image classification model's predictions through physical perturbations of images before they are recorded by a camera and pre-processed. It should also be noted that this characterization of an adversary's strength significantly differs from bounds on available computational resources as is commonly employed in various areas of security and cryptography. As the field of adversarial machine learning matures, we expect the research community to reflect and expand on this preliminary measure of strength limited to the adversary's knowledge of a system's potential weakness(es).

Moreover, adversarial incentives may define a range of attack goals. When the adversary is interested in having a given input classified in any class that is different from its correct class, we refer to this as an "untargeted misclassification attack." Instead, when the adversary is interested in defining the target class in which it intends to have the model misclassify an input from any (correct) source class, this is a "source-target misclassification attack" (also called a "targeted attack" in the literature).

Poisoning the model. Note that an adversary may also attack the training process itself. If the adversary can insert its own samples or otherwise corrupt the data used for training, the adversary can thus influence the model to induce incorrect connections between input features and classes (called "false learning") or reduce confidence in the labeling, decreasing model accuracy. In either case, the corruption of the training process compromises the integrity of the model and decreases its robustness against adversarial inputs. Although such attacks are being studied and countermeasures considered by the research community, they are beyond the scope of this article.

Adversarial Example Generation

The challenge for the adversary is figuring out how to generate an input with the desired output, as in the source-target-misclassification attack. In such an attack, the adversary starts with a sample that is legitimate (such as a STOP sign) and modifies it through a perturbation process to attempt to cause the model to classify it in a chosen target class (such as the one corresponding to a YIELD sign).

For an attack to be worth studying from a machine learning point of view, it is necessary to impose constraints that ensure the adversary is not able to truly change the class of the input. For example, if the adversary could physically replace a stop sign with a yield sign or physically paint a yield symbol onto a stop sign, a machine learning algorithm must be able to still recognize it as a yield sign. In the context of computer vision, we generally consider only modifications of an object's appearance that do not interfere with a human observer's ability to recognize the object. The search for misclassified inputs is thus done with the constraint that these inputs should be visually very similar to a legitimate input. Consider the two images in Figure 5, potentially consumed by an autonomous vehicle. To the human eye, they appear to be the same, and our biological classifiers (vision) identify each one as a stop sign. The image on the left is indeed an ordinary image of a stop sign. We produced the image on the right by adding a small, precise perturbation that forces a particular image classification deep neural network to classify it as a yield sign. Here, the adversary could potentially use the altered image to cause the car to behave dangerously, if the car lacks fail-safes (such as maps of known stop-sign locations). In other application domains, the constraint differs. When targeting machine learning models used for malware detection, the constraint becomes that the input—or malware software—misclassified by the model must still be in a legitimate executable format and execute its malicious logic when executed.[10]

White-box vs. black-box. One way to characterize an adversary's strength is the amount of access the adversary has to the model. In a white-box scenario, the adversary has full access to the model whereby the adversary knows what machine learning algorithm is being used and the values of the model's parameters. In this case, we show in the following paragraphs that constructing an adversarial example can be formulated as a straightforward optimization problem. In a black-box scenario, the attacker must rely on guesswork, because the machine learning algorithm used by the defender and the parameters of the defender's model are not known. Even in this scenario, where the attacker's strength is limited by incomplete knowledge, the attacker might still succeed. We now describe the white-box techniques first because they form the basis for the more difficult black-box attacks.

White-box attacks. An adversarial example x^* is found by perturbing an originally correctly classified input x. To find x^*, one solves a constrained optimization problem. One very generic approach, applicable to essentially all machine learning paradigms, is to solve for the x^* that causes the most expected loss (a metric reflecting the model's error), subject to a constraint on the maximum allowable deviation from the original input x; in the case of machine learning models solving classification tasks, the loss of a model can be understood as its prediction error. Another approach, specialized to classifiers, is to impose a constraint that the perturbation must cause a misclassification and solve for the smallest possible perturbation

$$x^* = x + \arg\min\{\|z\| : f(x+z) = t\} \quad (1)$$

where x is an input originally correctly classified, $\|\cdot\|$ a norm that appropriately quantifies the similarity constraints discussed earlier, and t is the target class chosen by the adversary. In the case of "untargeted attacks," t can be any class different from the correct class $f(x)$. For example, for an image the adversary might use the ℓ_0 "norm" to force the attack to modify very few pixels, or the ℓ_∞ norm to force the attack to make only very small changes to each pixel. All of these different ways of formulating the optimization problem search for an x^* that should be classified the same as x (because it is very similar to x) yet is classified differently by the model. These optimization problems are typically intractable, so most adversarial example-generation algorithms are based on tractable approximations.

Gradient-based search algorithms. The optimization algorithms described earlier are, in principle, intractable for most interesting models, because most interesting models use nonlinear, non-convex functions. In practice, gradient-based optimization algorithms reliably find solutions that cause misclassification, presumably because a point x^* can cause misclassification without being an optimal solution to Equation (1).

Several approaches using gradient-based optimization have been introduced to date. We present here three canonical examples of gradient-based attacks: the L-BFGS approach, the Fast Gradient Sign Method (FGSM), and the Jacobian Saliency Map Approach (JSMA).

Szegedy et al.[30] adapted the L-BFGS method to the constrained optimization problem outlined earlier. They were the first researchers to demonstrate perturbations indistinguishable from human observers that were sufficient to force a computer-vision model to misclassify an image encoded by x^*. Recently, the L-BFGS method was revisited by Carlini et al.[4] who found that using the Adam optimizer along with customized objectives reduces the size of the perturbation required to ensure an input will be misclassified. However, both approaches come at a high computational cost. It is possible to reduce that cost, usually at the price of also reducing the effectiveness of the attack. In general, attacks exist along a continuum. For example, it is possible to simply run the L-BFGS algorithm for fewer iterations to obtain a less-expensive attack with a lower success rate.

The adversary's strength is characterized by its ability to access the model's architecture, parameter values, and training data.

One attack with especially low computational cost is the FGSM,[9] an approach that maximizes the model's prediction error while keeping the ℓ_0 norm of the perturbation added to the input constant. This attack is based on the observation that many machine learning models, even neural networks, are very linear as a function of the input x. One way to formulate an adversarial attack is

$$x^* = \max_{x^*} J_f(x) \text{subject to} \|x^* - x\|_\infty \leq \epsilon, (2)$$

where J_f is the expected loss incurred by the machine learning model, and is a way to measure the model's prediction error. This optimization algorithm is typically intractable, but if the true J_f is replaced with a first-order Taylor series approximation of J_f formed by taking the gradient at x, the optimization problem can be solved in closed form

$$x^* = x + \epsilon \cdot sign(\nabla_x J_f(x, y)) \quad (3)$$

Because the linear approximation used by the Taylor series expansion approximately holds, it often finds adversarial examples despite its low runtime requirements.

In the JSMA, the adversary chooses a target class in which the sample should be misclassified by the model.[22] Given model f, the adversary crafts an adversarial sample $x^* = x + \delta_x$ by adding a perturbation δ_x to a subset of the input components x_i. To choose the perturbation, the adversary would sort the components by decreasing adversarial saliency value. Intuitively, a saliency value is a measure of how important a particular feature is to determining a given output class (such as the importance of a particular pixel or group of pixels in determining what kind of sign is in an image). The adversarial saliency value $S(f, x, t)[i]$ of component i for an adversarial target class t is defined as

$$S(f,x,t)[i] = \begin{cases} 0 \text{ if } \frac{\partial f_t}{\partial x_i}(x) < 0 \text{ or } \sum_{j\neq t} \frac{\partial f_j}{\partial x_i}(x) > 0 \\ \frac{\partial f_t}{\partial x_i}(x) \left| \sum_{j\neq t} \frac{\partial f_j}{\partial x_i}(x) \right| \text{otherwise} \end{cases} \quad (4)$$

where matrix

$$J_f = \left[\frac{\partial f_j}{\partial x_i}\right]_{ij}$$

is the model's Jacobian matrix. Input components i are added to perturbation δ_x in order of decreasing adversarial saliency value $S(x, t)[i]$ until the resulting adversarial sample $x^* = x + \delta_x$ is misclassified by f. The perturbation introduced for each selected input component can vary, and larger perturbations usually mean that fewer components need to be perturbed.

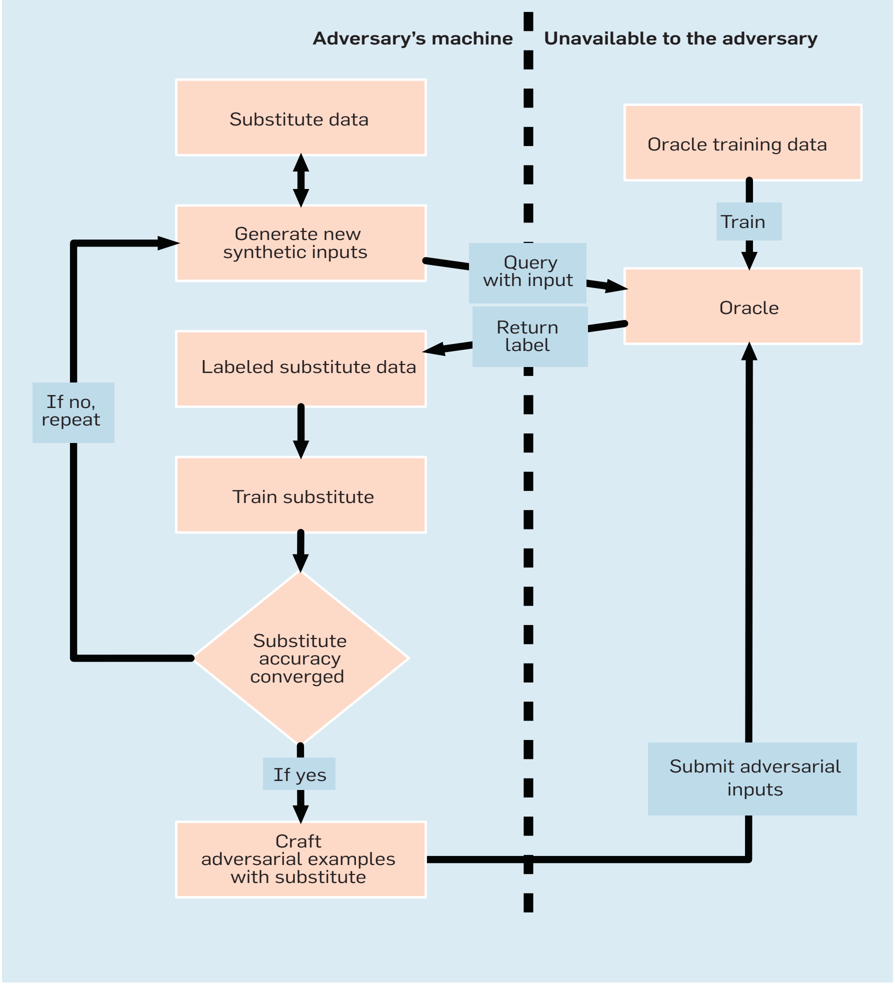
Each algorithm has its own benefits and drawbacks. The FGSM is well suited for fast crafting of many adversarial examples with relatively large perturbations and is potentially easier to detect. JSMA and L-BFGS (or other iterative optimization algorithms) both produce stealthier perturbations at greater computational cost. FGSM sometimes works better than L-BFGS if the gradient is very small, because the sign operation removes the dependence on the gradient magnitude. All of these algorithms can fail to fool the classifier. On typical machine learning benchmark problems, all three algorithms have a near-100% success rate against normal machine learning algorithms. Defense techniques can thwart a high percentage of FGSM and JSMA attacks, but L-BFGS is essentially a brute-force white-box approach that almost always succeeds regardless of defense techniques, given enough runtime.

Black-box attacks. Although in some cases it is possible for the adversary to have access to a model's parameters, in most realistic threat models, the adversary has access to the model only through a limited interface. Additional strategies are thus required to conduct attacks without access to the model's gradients, which are unavailable in a limited-interface black-box scenario.

In such threat models, a possible strategy for the adversary is to train another model, different from the one being targeted. This model, called the "substitute," is used by an adversary to compute the gradients required for the attack. Assuming that both the substitute and the targeted models operate in similar ways (such as they have a similar decision boundary), an adversarial example computed using the substitute model is highly likely to be misclassified by the targeted one. This transfer of adversarial examples from a substitute model to a target model was first demonstrated by Szegedy et al.[30]

There are two ways to pursue such a strategy. One is for the attacker to collect and label the attacker's own training set.[30] This approach works with absolutely no access to the model but can be expensive because it might require gathering a large number of real input examples and human effort to label each example. When the attacker is able to query the target model by sending it inputs and observing the returned outputs, a much less-expensive approach is to strategically send algorithmically generated inputs in order to reverse engineer the target model, without any (or very little) training data. Such a strategy was introduced by Papernot et al.[21] and is illustrated in Figure 6. In this work, the adversary has access only to the model through an API that returns the class predicted by the model for any input chosen by the adversary. That is, the API acts as an "oracle" for the model. In addition to the model architecture and training data being unavailable to the adversary, the adversary does not even need to know what type of architecture is used to create the machine learning model, as if the model is, say, a support vector machine, a neural network, or something else. Through interactions with the API, the adversary is going to create a training set for its substitute model, in a way that ensures the substitute makes predictions similar to the one

Figure 6. Black-box attack strategy introduced by Papernot et al.[21]


Adversary's machine
Unavailable to the adversary
Substitute data
Oracle training data
Generate new synthetic inputs
Train
Query with input
Oracle
Return label
Labeled substitute data
If no, repeat
Train substitute
Substitute accuracy converged
If yes
Submit adversarial inputs
Craft adversarial examples with substitute

made by the targeted model and is, in turn, a good indicator of where the targeted model will make mistakes. The idea is that the adversary uses the oracle to explore and reconstruct the decision boundary of the victim model. The key is being intelligent about that exploration.

The main challenge resides in selecting the inputs the adversary sends to the API to reduce the total number of queries made—to reduce the detectability of the attack. The approach taken by Papernot et al.[21] is to generate synthetic inputs using a few real inputs collected by the adversary. For instance, when targeting a traffic-sign classifier, the adversary would collect a small set of images of each traffic sign and then run the following augmentation technique for each of these images to find new inputs that should be labeled with the API. The augmentation technique takes a set of training inputs S_p, the labels they were given $\forall x \in S_p$, $f(x)$ by the targeted model f, and the current substitute model g

$$S_{\rho+1} \leftarrow \{x + \lambda \cdot sign(J_g[f(\vec{x})]) : x \in S_\rho\} \cup S_\rho \quad (5)$$

where J_g denotes here the Jacobian matrix of g. The substitute is trained by successively labeling and augmenting S_p.

When the substitute is sufficiently accurate, the adversary can use it to run one of the attacks described earlier, as in, for instance, the FGSM or the JSMA. The adversarial examples produced by these algorithms on the substitute g are likely to transfer to the targeted model f. This "transferability" property, first observed among deep neural networks and linear models by Szegedy et al.,[30] is known to hold across many types of machine learning models.[20] Indeed, Figure 7 reports transferability rates, the number of adversarial examples misclassified by a model B, despite being crafted with a different model A, for several pairs of machine learning models trained on a standard image-recognition benchmark as in the MNIST dataset of handwritten digits. For many such pairs, transferability is generally consistent between the two models. Adversarial examples even transfer to an ensemble of models that make predictions based on a majority vote among the class predicted by a collection of independent models.

Black-box attacks using these strategies have been demonstrated against proprietary models and accessible through a "machine learning as a service" query APIs (such as those from Google and Amazon[21,33]).

Yet other strategies exist for black-box attacks, with more identified every day. Notably, Xu et al.[35] found that genetic algorithms produce adversarial examples. However, this requires that the adversary is able to access the output confidence values (such as output probabilities) returned by the model or at least to approximate these confidence values.[5] These probabilities are used to compute a fitness function for genetic variants of the original input, and, in turn, retain the variants that are most "fit" to achieve the adversarial goal of misclassification. A drawback of this strategy is that, compared to other black-box strategies,[21,30] the algorithm must be able to make a large number of model-prediction queries before it finds evasive variants and is thus more likely to be detected at runtime.

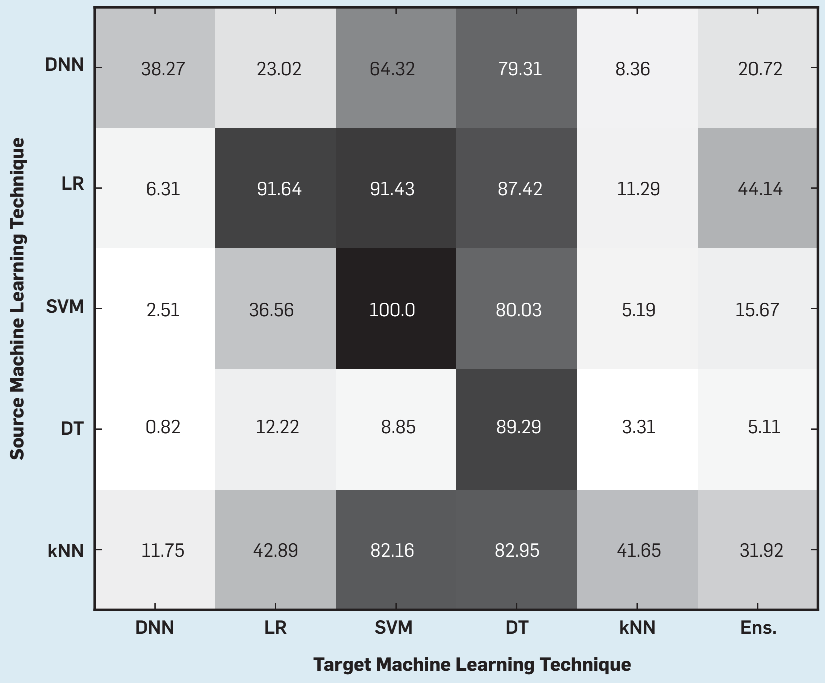
Defenses and Their Limitations

Given the existence of adversarial samples, an important question is: What can be done to defend models against them? The defensive property of a machine learning system in this context is called "robustness against adversarial samples." We now explore several known defenses categorized into three classes: model training, input validation, and architectural changes.

Model training. Adversarial training and defensive distillation are two techniques proposed to train models that explicitly attempt to be robust to adversarial examples. Adversarial training seeks to improve the generalization of a model when presented with adversarial examples at test time by proactively generating adversarial examples as part of the training procedure. This idea was first introduced by Szegedy et al.[30] but not yet practical due to the high computational cost of generating adversarial examples. Goodfellow et al.[9] then showed how to generate adversarial examples inex-

Figure 7. Transferability matrix. The source model is used to craft adversarial examples and the target model to predict their class.

The transferability reported is the average rate of mistakes made by the target model on adversarial examples crafted using the source model. The models considered are a deep neural network, logistic regression, support vector machine, decision tree, and *k*-nearest neighbor. The ensemble target model refers to an ensemble making predictions based on a majority vote among the class predicted by each of the five previous models. All adversarial examples were produced with similar perturbation norms.[20]



pensively with the fast-gradient-sign method and made it computationally efficient to continuously generate new adversarial examples every time the model parameters change during the training process. The model is then trained to assign the same label to the adversarial example as to the original example.

Defensive distillation smooths the model's decision surface in adversarial directions that could be exploited by the adversary.[24] Distillation is a training procedure whereby one model is trained to predict the probabilities output by another model that was trained earlier. Distillation was introduced by Hinton et al.[11] aiming for a small model to mimic a large, computationally expensive model. Defensive distillation has a different goal—make the final model's responses more smooth—so it works even if both models are the same size. It may seem counterintuitive to train one model to predict the output of another model with the same architecture. The reason it works is that the first model is trained with "hard" labels (100% probability an image is a STOP sign rather than a YIELD sign) and then provides "soft" labels (95% probability an image is a STOP sign rather than a YIELD sign) used to train the second model. The second distilled model is more robust to attacks (such as the fast gradient sign method and the Jacobian-based saliency map approach).

Both adversarial training and defensive distillation suffer from limitations, however. They are generally effective against inexpensive white-box attacks but can be broken using black-box attacks[21] or computationally expensive attacks based on iterative optimization.[4] These two strategies are examples of defenses that perform gradient masking,[21,23] removing the gradient of the model used by the adversary to find good-candidate directions to construct adversarial examples. However, the adversary can still evade the model practically if it can find other ways to identify candidate adversarial directions (such as through a black-box attack) or start the gradient-based search outside the input region impacted by the defense (such as by first taking a step in a random direction[32]).

Input validation and preprocessing. Perhaps the most obvious defense is to validate the input before it is given to the model and possibly preprocess it to remove potentially adversarial perturbations. In many application domains there are verifiable properties of inputs that should never be violated in practice. For example, an input image from a camera sensor can be checked for realism; for example, certain properties of cameras and light ensure certain pixel neighborhoods (such as neighbor pixels with exceptionally high contrast) never occur. Such defenses are limited in that they are highly domain dependent and subject to environmental factors. Moreover, it is not clear that the constraints placed on the domain just increase the difficulty of adversarial sample generation or provide broad protections. It is highly likely that the effectiveness of input constraints as a countermeasure is also domain specific.

Architecture modifications. It is also possible to defend against adversarial samples by altering the structure of the machine learning system.

Highly nonlinear machine learning models are more robust to adversarial examples but also more difficult to train and generally do not perform well in a baseline non-adversarial setting.[9]

Future work may reveal architectures that resist adversarial examples yet are amenable to effective training.

Future work may reveal architectures that resist adversarial examples yet are amenable to effective training.

From Testing to Verification

The limitations of existing defenses point to the lack of theory and practice of verification and testing of machine learning models. To design reliable systems, engineers engage in both testing and verification. By testing, we mean evaluating the system under various conditions and observing its behavior, watching for defects. By "verification," we mean producing a compelling argument that the system will not misbehave under a broad range of circumstances.

Machine learning practitioners have traditionally relied primarily on testing. A classifier is usually evaluated by applying the classifier to several examples drawn from a test set and measuring its accuracy on these examples.

To provide security guarantees, it is necessary to ensure properties of the model besides its accuracy on naturally occurring test-set examples. One well-studied property is robustness to adversarial examples. The natural way to test robustness to adversarial examples is simply to evaluate the accuracy of the model on a test set that has been adversarially perturbed to create adversarial examples.[30]

Unfortunately, testing is insufficient for providing security guarantees, as an attacker can send inputs that differ from the inputs used for the testing process. For example, a model that is tested and found to be robust against the fast gradient sign method of adversarial example generation[9] may be vulnerable to computationally expensive methods like attacks based on L-BFGS.[30]

In general, testing is insufficient because it provides a "lower bound" on the failure rate of the system when an "upper bound" is necessary for providing security guarantees. Testing identifies *n* inputs that cause failure, so the engineer can conclude that at least *n* inputs cause failure; the engineer would prefer to have a means of becoming reasonably confident that at most *n* inputs cause failure.

Putting this in terms of security, a defense should provide a measurable guarantee that characterizes the space of inputs that cause failures. Conversely, the common practice of testing can only provide instances that cause error and is thus of limited value in understanding the robustness of a machine learning system. Development of an input-characterizing guarantee is central to the future of machine learning in adversarial settings and will almost certainly be grounded in formal verification.

Theoretical verification of machine learning. Verification of machine learning model robustness to adversarial examples is in its infancy. Current approaches verify that a classifier assigns the same class to all points within a specified neighborhood of a point x. Huang et al.[13] developed the first verification system for demonstrating that the output class is constant across a desired neighborhood. This first system uses an SMT solver. Its scalability is limited, and scaling to large models requires making strong assumptions (such as that only a subset of the units of the network is relevant to the classification of a particular input point). Such assumptions mean the system can no longer provide an absolute guarantee of the absence of adversarial examples, as an adversarial example that violates the assumptions could evade detection. Reluplex[15] is another verification system that uses linear programming solvers to scale to much larger networks. Reluplex is able to become much more efficient by specializing on rectified linear networks[6,14,18] and their piecewise linear structure.

These current verification systems are limited in scope because they verify only that the output class remains constant in some specified neighborhood of some specific point x. It is infeasible for a defender to fully anticipate all future attacks when specifying the neighborhood surrounding x to verify. For instance, a defender may use a verification system to prove there are no adversarial examples within a max-norm ball of radius $\in$, but then an attacker may devise a new way of modifying x that should leave the class unchanged yet has a high max-norm. An even greater challenge is verifying the behavior of the system near new test points x'.

In a traditional machine learning setting there are clear theoretical limits as to how well a machine learning system can be expected to perform on new test points. For example, the "no free lunch theorem"[34] states that all supervised classification algorithms have the same accuracy on new test points when averaged over all possible datasets.

One important open theoretical question is: Can the no-free-lunch theorem be extended to the adversarial setting? If we assume attackers operate by making small perturbations to the test set, then the premise of the no-free-lunch theorem, where the average is taken over all possible datasets, including those with small perturbations, should not be ignored by the classifier, no longer applies. Depending on the resolution of this question, the arms race between attackers and defenders could have two different outcomes. The attacker might fundamentally have the advantage due to inherent statistical difficulties associated with predicting the correct value for new test points. If we are fortunate, the defender might have a fundamental advantage for a broad set of problem classes, paving the way for the design and verification of algorithms with robustness guarantees.

Reproducible testing with `CleverHans`. While verification is a challenge even from a theoretical point of view, straightforward testing can likewise be a challenge from a practical point of view. Suppose a researcher proposes a new defense procedure and evaluates that defense against a particular adversarial example attack procedure. If the resulting model obtains high accuracy, does it mean the defense was effective? Possibly, but it could also mean the researcher's implementation of the attack was weak. A similar problem occurs when researchers test a proposed attack technique against their own implementation of a common defense procedure.

To resolve these difficulties, we created the `CleverHans`[19] library with reference implementations of several attack and defense procedures. Researchers and product developers can use `CleverHans` to test their models against standardized, state-of-the-art attacks and defenses. This way, if a defense obtains high accuracy against a `CleverHans` attack, the test would show that the defense overcomes this standard implementation of the attack, and if an attack obtains a high failure rate against a `CleverHans` defense, the test would show that the attack is able to defeat a rigorous implementation of the defense. While such standardized testing of attacks and defenses does not substitute in any way to rigorous verification, it does provide a common benchmark. Moreover, results in published research are comparable to one another, so long as they are produced with the same version of `CleverHans` in similar computing environments.

Future of Adversarial Machine Learning

Adversarial machine learning is at a turning point. In the context of adversarial inputs at test time, we have several effective attack algorithms but few strong countermeasures. Can we expect this situation to continue indefinitely? Can we expect an arms race with attackers and defenders repeatedly seizing the upper hand in turn? Or can we expect the defender to eventually gain a fundamental advantage?

We can explain adversarial examples in current machine learning models as the result of unreasonably linear extrapolation[9] but do not know what will happen when we fix this particular problem; it may simply be replaced by another equally vexing category of vulnerabilities. The vastness of the set of all possible inputs to a machine learning model seems to be cause for pessimism. Even for a relatively small binary vector, there are far more possible input vectors than there are atoms in the universe, and it seems highly improbable that a machine learning algorithm would be able to process all of them acceptably. On the other hand, one may hope that as classifiers become more robust, it could become impractical for an attacker to find input points that are reliably misclassified by the target model, particularly in the black-box setting.

These questions may be addressed empirically, by actually playing out the arms race as new attacks and new countermeasures are developed. We may also be able to address these questions theoretically, by proving the arms race must converge to some asymptote. All these endeavors are difficult, and we hope many will be inspired to join the effort.

Acknowledgments
Author Nicolas Papernot is supported by a Google Ph.D. Fellowship in Security. Research was supported in part by the Army Research Laboratory under Cooperative Agreement Number W911NF-13-2-0045 (ARL Cyber Security CRA) and the Army Research Office under grant W911NF-13-1-0421. The views and conclusions contained in this article are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. government. The U.S. government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright notation hereon. C

References
1. Arp, D., Spreitzenbarth, M., Hubner, M., Gascon, H., Rieck, K., and Siemens, C.E.R.T. Drebin: Effective and explainable detection of Android malware in your pocket. In *Proceedings of the NDSS Symposium* (San Diego, CA, Feb.). Internet Society, Reston, VA, 2014, 23–26.
2. Barreno, M., Nelson, B., Sears, R., Joseph, A.D., and Tygar, J.D. Can machine learning be secure? In *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security* (Taipei, Taiwan, Mar. 21–24). ACM Press, New York, 2006, 16–25.
3. Bolton, R.J. and Hand, D.J. Statistical fraud detection: A review. *Statistical Science 17*, 3 (2002), 235–249.
4. Carlini, N. and Wagner, D. Towards evaluating the robustness of neural networks. *arXiv preprint*, 2016; https://arxiv.org/pdf/1608.04644.pdf
5. Dang, H., Yue, H., and Chang, E.C. Evading classifier in the dark: Guiding unpredictable morphing using binary-output blackboxes. *arXiv preprint*, 2017; https://arxiv.org/pdf/1705.07535.pdf
6. Glorot, X., Bordes, A., and Bengio, Y. Deep sparse rectifier neural networks. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics* (Ft. Lauderdale, FL, Apr. 11–13, 2011), 315-323.
7. Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, Cambridge, MA, 2016; http://www.deeplearningbook.org/
8. Goodfellow, I.J., Bulatov, Y., Ibarz, J., Arnoud, S., and Shet, V. Multi-digit number recognition from Street View imagery using deep convolutional neural networks. In *Proceedings of the International Conference on Learning Representations* (Banff, Canada, Apr. 14–16, 2014).
9. Goodfellow, I.J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *arXiv preprint*, 2014; https://arxiv.org/pdf/1412.6572.pdf
10. Grosse, K., Papernot, N., Manoharan, P., Backes, M., and McDaniel, P. Adversarial perturbations against deep neural networks for malware classification. In *Proceedings of the European Symposium on Research in Computer Security* (Oslo, Norway, 2017).
11. Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *arXiv preprint*, 2015; https://arxiv.org/abs/1503.02531
12. Huang, S., Papernot, N., Goodfellow, I., Duan, Y., and Abbeel, P. Adversarial attacks on neural network policies. *arXiv preprint*, 2017; https://arxiv.org/abs/1702.02284
13. Huang, A., Kwiatkowska, M., Wang, S., and Wu, M. Safety verification of deep neural networks. In *Proceedings of the International Conference on Computer-Aided Verification* (2016); https://link.springer.com/chapter/10.1007/978-3-319-63387-9_1
14. Jarrett, K., Kavukcuoglu, K., Ranzato, M.A., and LeCun, Y. What is the best multi-stage architecture for object recognition? In *Proceedings of the 12th IEEE International Conference on Computer Vision* (Kyoto, Japan, Sept. 27–Oct. 4). IEEE Press, 2009.
15. Katz, G., Barrett, C., Dill, D., Julian, K., and Kochenderfer, M. Reluplex: An efficient SMT solver for verifying deep neural networks. In *Proceedings of the International Conference on Computer-Aided Verification*. Springer, Cham, 2017, 97–117.
16. Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial examples in the physical world. In *Proceedings of the International Conference on Learning Representations* (2017); https://arxiv.org/abs/1607.02533
17. Murphy, K.P. *Machine Learning: A Probabilistic Perspective*. MIT Press, Cambridge, MA, 2012.
18. Nair, V. and Hinton, G.E. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the International Conference on Machine Learning* (Haifa, Israel, June 21–24, 2010).
19. Papernot, N., Goodfellow, I., Sheatsley, R., Feinman, R., and McDaniel, P. CleverHans v2.1.0: An adversarial machine learning library; https://github.com/tensorflow/cleverhans
20. Papernot, N., McDaniel, P., and Goodfellow, I. Transferability in machine learning: From phenomena to black-box attacks using adversarial samples. *arXiv preprint*, 2016; https://arxiv.org/abs/1605.07277
21. Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z.B., and Swami, A. Practical black-box attacks against deep learning systems using adversarial examples. In *Proceedings of the ACM Asia Conference on Computer and Communications Security* (Abu Dhabi, UAE). ACM Press, New York, 2017.
22. Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z.B., and Swami, A. The limitations of deep learning in adversarial settings. In *Proceedings of the 2016 IEEE European Symposium on Security and Privacy* (Saarbrücken, Germany, Mar. 21–24). IEEE Press, 2016, 372–387.
23. Papernot, N., McDaniel, P., Sinha, A., and Wellman, M. Towards the science of security and privacy in machine learning. In *Proceedings of the Third IEEE European Symposium on Security and Privacy* (London, U.K.); https://arxiv.org/abs/1611.03814
24. Papernot, N., McDaniel, P., Wu, X., Jha, S., and Swami, A. Distillation as a defense to adversarial perturbations against deep neural networks. In *Proceedings of the 37th IEEE Symposium on Security and Privacy* (San Jose, CA, May 23–25). IEEE Press, 2016, 582–597.
25. Russell, S. and Norvig, P. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 1995, 25–27.
26. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M. et al. Mastering the game of Go with deep neural networks and tree search. *Nature 529*, 7587 (2016), 484–489.
27. Stallkamp, J., Schlipsing, M., Salmen, J., and Igel, C. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks* (2012); https://doi.org/10.1016/j.neunet.2012.02.016
28. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, C., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Press, 2015, 1–9.
29. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the Inception architecture for computer vision. *ArXiv e-prints*, Dec. 2015; https://arxiv.org/abs/1512.00567
30. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. In *Proceedings of the International Conference on Learning Representations*, 2014.
31. Taigman, Y., Yang, M., Ranzato, M.A., and Wolf, L. DeepFace: Closing the gap to human-level performance in face verification. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. IEEE Press, 2014.
32. Tramèr, F., Kurakin, A., Papernot, N., Boneh, D., and McDaniel, P. Ensemble adversarial training: Attacks and defenses. *arXiv preprint*, 2017; https://arxiv.org/abs/1705.07204
33. Tramèr, F., Zhang, F., Juels, A., Reiter, M.K., and Ristenpart, T. Stealing machine learning models via prediction APIs. In *Proceedings of the USENIX Security Conference* (San Francisco, CA, Jan. 25–27). USENIX Association, Berkeley, CA, 2016.
34. Wolpert, D.H. The lack of a priori distinctions between learning algorithms. *Neural Computation 8*, 7 (1996), 1341–1390.
35. Xu, W., Qi, Y., and Evans, D. Automatically evading classifiers. In *Proceedings of the 2016 Network and Distributed Systems Symposium* (San Diego, CA, Feb. 21–24). Internet Society, Reston, VA, 2016.

Ian Goodfellow (goodfellow@google.com) is a staff research scientist at Google Brain, Mountain View, CA, USA, and inventor of Generative Adversarial Networks.

Patrick McDaniel (mcdaniel@cse.psu.edu) is the William L. Weiss Professor of Information and Communications Technology in the School of Electrical Engineering and Computer Science at Pennsylvania State University, University Park, PA, USA, and a fellow of both IEEE and ACM.

Nicolas Papernot (ngp5056@cse.psu.edu) is a Google Ph.D. Fellow in Security in the Department of Computer Science and Engineering at Penn State University, University Park, PA, USA.

Watch the authors discuss their work in this exclusive *Communications* video. https://cacm.acm.org/videos/making-machine-learning-robust-against-adversarial-inputs